プログラミング言語論

プログラミング言語の構文

水野嘉明

目次

2

1. プログラミング言語の構文

- *構文*（Syntax）
  - プログラムの書き方を規制する文法規則
- 構文の記述
  - BNF ／ 文脈自由文法
  - 構文図式

3

1. プログラミング言語の構文

- 構文，制約，意味
  - 構文：プログラムの構造を定義
  - 制約：構文以外の規則（constraint）
  - 意味：プログラムの働きを定義（semantics）

4

1. プログラミング言語の構文

- 例：if文
  - 構文：**if**（*expr*） s_1 **else** s_2
  - 制約：*expr*は真偽値を表す式
  - 意味：*expr*を計算し、その値が真ならs_1を，偽ならs_2を実行する
- 文法：構文＋制約

5

2. BNF

- *BNF*（Backus Naur form）とは
  - 構文を記述するための表記法
  - 1959 バッカス(John Backus)が考案、ナウア(Peter Naur)が改良してAlgol の定義に採用
  - 文脈自由文法（後述）と同じ記述能力 （表記法が違うだけ）

6

2. BNF

- BNFは、構文を定義するだけ
  - 意味を定義するものではない

7

2. BNF

- BNFの例（1）
  - 識別子

```
<letter>::=a|b|c|…|z|A|B|…|X|Y|Z
<digit>::=0|1|2|3|4|5|6|7|8|9
<identifier>::=<letter>
             |<identifier><letter>
             |<identifier><digit>
```

8

2. BNF

- この例は、
  - <letter>とは、a～z 、A～Zの1字
  - <digit>とは、0～9の1字
  - <identifier>とは、<letter> または <identifier>の後に<letter>か <digit>を続けたもの

と、定義している

9

2. BNF

- これは、

  識別子は英数字の並びである。
  ただし1文字目は英字でなければならない。

  という一般的な定義を、きちんと定義したもの

10

2. BNF

- BNFの文法
  - ::= は、左辺が右辺により定義されることを表す
    (「～とは」と読むとよい)
  - 縦棒 | は、選択を表す
    (「または」)

11

2. BNF

- <XX>は、プログラム中には直接現れることはない、抽象的な名前

  非終端記号 (Nonterminal Symbols)
- 1、2、a、b、+、*、(、) 等 プログラムに出現する記号 (それ以上書き換えできない記号)が

  終端記号 (Terminal Symbols)

12

2. BNF

- BNFの例（2）
  - 演算子＋,＊を用いた式の構文規則

<E>::= <E>+<E>|<E>*<E>|num|(<E>)

  - num は式である
  - xと yが式であれば、x+y、x*y、および (x)も式である
  - 上記(1)(2)で定義されたものだけが式である

13

2. BNF

- 演習4.1

次のBNFで定義されるビット列Sであるものはどれか

<S> ::= 01 | 0<S>1

ア. 000111　　イ. 010010
ウ. 010101　　エ. 011111

（基本情報技術者 平20春 午前 問11）

14

3. 文脈自由文法

- *文脈自由文法*
  （Context Free Grammar）
  - 形式文法の1つ
  - 特に、プログラミング言語の構文仕様の定義において、重要

15

3. 文脈自由文法

- 一般的な形式文法の構成要素
  - *終端記号* (Terminal Symbols)の有限集合
    - その言語で使われる文字
    - それ以上書き換えできない記号

16

3. 文脈自由文法

- *非終端記号* (Nonterminal Symbols)の有限集合
  - プログラム中には直接現れることはない、抽象的な名前
  - BNFでは ＜＞で囲んで示すが、ここでは、全てを列挙する
  - 終端記号と共通の元を含まない

17

3. 文脈自由文法

- *生成規則* (Production Rules)の有限集合
  - 非終端記号を含む記号列を書き換えるための規則
  - $\alpha \rightarrow \beta$ という形式
    ただし、α、β は終端記号や非終端記号の列

18

3. 文脈自由文法

- *開始記号* (Start Symbol)
  - 書き換えを始める最初の非終端記号
  - 開始記号から始めて生成規則を適用していくことによって、終端記号のみから構成される単語を生成する

19

3. 文脈自由文法

- 文脈自由文法Gの定義

  $G = (N, \Sigma, P, S)$

  ここで、

  N: 非終端記号の有限集合

  Σ: 終端記号の有限集合

  P: 生成規則$A \rightarrow \alpha$の有限集合

  ただし $A \in N, \alpha \in (N \cup \Sigma)^*$

  S: 開始記号 ($S \in N$)

20

3. 文脈自由文法

- 導出
  - 生成規則 $A \rightarrow \alpha \in P$と記号列$\beta$、$\gamma \in (N \cup \Sigma)^*$ がある時、$\beta A \gamma$は$\beta \alpha \gamma$に変換できる

    $\beta A \gamma \Rightarrow \beta \alpha \gamma$

  これを *導出* (derivation) という

21

3. 文脈自由文法

- 導出 $\beta A \gamma \Rightarrow \beta \alpha \gamma$ において、記号列 β、$\gamma \in (N \cup \Sigma)^*$ を *文脈* という
  - 文脈が何であっても導出に変わりがないとき、*文脈自由* であるという

22

3. 文脈自由文法

- $\alpha_0 \Rightarrow \alpha_1$、$\alpha_1 \Rightarrow \alpha_2$、$\alpha_2 \Rightarrow \alpha_3$、・・・、$\alpha_{n-1} \Rightarrow \alpha_n$ の時

「α_n は α_0 から n回の導出により得られる」という

$$\alpha_0 \Rightarrow^* \alpha_n$$

注:$\Rightarrow^*$ は $\Rightarrow$の反射的推移閉包

23

3. 文脈自由文法

- 文脈自由文法では、開始記号から始まり生成規則にしたがって書き換え(導出)を繰り返す
- 記号が全て終端記号になった時点で、終了する

24

3. 文脈自由文法

- BNFは、文脈自由文法とまったく同じ
  - 表記法は、異なる
    - 文脈自由文法では、::=の代わりに→を用いる （生成規則）
    - BNFでは、< >により非終端記号と終端記号を区別する
      文脈自由文法では、宣言により区別する

25

3. 文脈自由文法

- 文脈自由文法で記述できる言語を *文脈自由言語*（Context Free Language）と呼ぶ
  - 文脈自由文法Gが生成する言語（終端記号列の集合）を L(G) と書く

$$L(G) = \{\omega \in \Sigma^* \mid S \Rightarrow^* \omega\}$$

26

3. 文脈自由文法

「a^nとは aをnケ並べたもの」という意味

- 文脈自由言語の例 (1)

  $L = \{ a^n b^n \mid n \geqq 1 \}$ とする

  ただし $a^n = \begin{cases} a & (n=1 \text{ の時}) \\ a^{n-1}a & (n>1 \text{ の時}) \end{cases}$

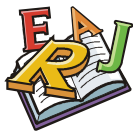
  Lを生成する文脈自由文法Gは

$$G = (N = \{S\},\ \Sigma = \{a,b\},\ P = \{S \rightarrow aSb,\ S \rightarrow ab\},\ S)$$

27

3. 文脈自由文法

- 「ab」「aaabbb」「aaaaaabbbbbb」などが、この言語 L(G)の *語* (word) である



28

3. 文脈自由文法

- 文脈自由言語の例 (2)
  次の文法G_2は、C言語の構文的に正しい文の集合の一部を生成する
  - $G_2 = (N, \Sigma, P, S)$
    N = { S, E, C }
    Σ = { id, num, +, (,), if, '{', '}', ;, >, = }
    P = { S→id=E; | SS | '{'S'}' | if(C)S,
    C→E>E,
    E→num | id | E+E }

29

3. 文脈自由文法

- この文法で生成される文の例
  - id=num;
  - {id=id+num;id=num;}
  - if(id>id+num)id=num;
- 例えば、x=y+3; に対応する終端記号列は id=id+num; である

30

3. 文脈自由文法

- if(id>id+num)id=num;
  の導出
  S⇒if(C)S
  ⇒if(E>E)S
  ⇒if(id>E)S
  ⇒if(id>E+E)S
  ⇒if(id>id+E)S
  （続く）

31

3. 文脈自由文法

（続き）
⇒if(id>id+num)S
⇒if(id>id+num)id=E;
⇒if(id>id+num)id=num;

※この文は、例えば次式に対応する
if (x>y+1) y=10;

32

3. 文脈自由文法

- 演習4.2
  $L_1 = \{ a^n b^m \mid n \geqq m \geqq 1 \}$とする

  L_1を生成する文脈自由文法G_1を求めよ

33

3. 文脈自由文法

- 演習4.3

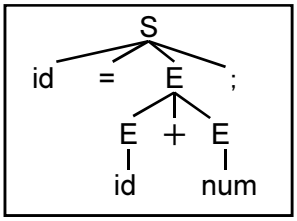
  $L_2 = \{ a^n b^m c^m, a^n b^n c^m \mid n, m \geqq 1 \}$ とする

  L_2を生成する文脈自由文法G_2を求めよ

34

3. 文脈自由文法

- *導出木* (derivation tree)
  - ➢導出過程を木構造で表現したもの

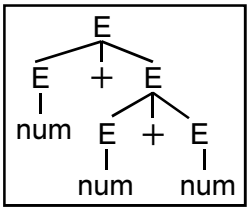


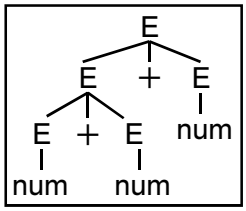
id=id+num;
という式の導出を表す

35

3. 文脈自由文法

- ➢一つの終端記号列に対し、導出や導出木は一般に複数存在する





36

3. 文脈自由文法

- 一つの導出木に対する導出は複数存在する
  - 非終端記号を展開する順序は複数ある
  - *最左導出* (left most derivation)
    - 最も左にある非終端記号から展開
  - *最右導出* (right most derivation)
    - 最も右にある非終端記号から展開

37

3. 文脈自由文法

- 最左導出の例
  - P.31～P.32 の導出例は、最左導出
- 一つの導出木に対する最左導出・最右導出は、各々一つだけ

38

3. 文脈自由文法

- 演習4.4

  演習4.2の言語 L_2

  $L_2 = \{ a^n b^m c^m, a^n b^n c^m \mid n, m \geqq 1 \}$

  について、終端記号列 abc の導出木をすべて求めよ

39

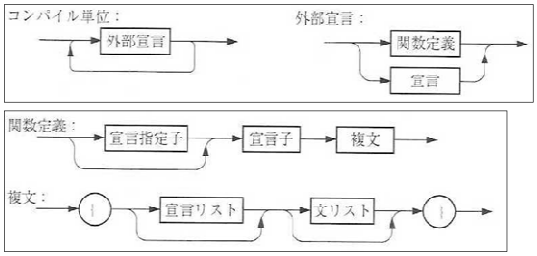
4. 構文図式

- *構文図式*（syntax diagram）とは
  - 構文を図で示す方法
  - Pascalの定義に用いられた
  - 簡潔で理解しやすい

40

4. 構文図式

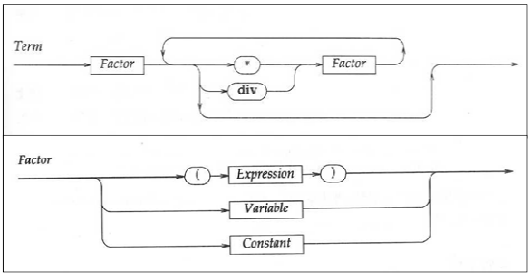
- 例1（C言語の構文規則 部分）



41

4. 構文図式

- 例2 （Pascalの構文規則 部分）



42

4. 構文図式

- 非終端記号は四角で、終端記号は丸で囲まれる
- 構文図式は、文脈自由文法の生成規則の右辺に正規表現を許した拡張文脈自由文法の図式表現である

43

【参考1】 チョムスキー階層

- 形式文法を、生成される言語の複雑さで分類した階層
  - ノーム・チョムスキーが提案（1956）
  - 0型～3型　の4タイプ

44

【参考1】 チョムスキー階層

型	文法	制限
0型	句構造文法	なし
1型	文脈依存文法	$\beta A \gamma \rightarrow \beta \alpha \gamma$
2型	文脈自由文法	$A \rightarrow \alpha$
3型	正規文法	A→a | A→aB | A→Ba

$\alpha, \beta, \gamma \in (N \cup \Sigma)^*$、A,B∈N、a∈Σ

45

【参考2】閉包

- 集合の*閉包*

P.20の　$\alpha \in (N \cup \Sigma)^*$　とは、

αは、非終端記号（Nの元）や 終端記号（Σの 元）を 0個以上並べたものである

という意味である

46

【参考3】反射的推移閉包

- 導出の*反射的推移閉包*

P.23の ⇒* は 関係 ⇒ （導出）の反射的推移閉包である

A ⇒* α とは、「Aから0回以上の導出を繰り返せば、αにたどり着く」ということを表している

47

お疲れ様でした